PROJECT SEMESTER REPORT

AUTOMATIC NUMBER PLATE RECOGNITION

SUBMITTED BY

Name:	Roll no:
1. **Chiraag Garg**	**UE198031**
2. **Deep Agam Kaur**	**UE198032**
3. **Deepak Goyal**	**UE198033**

UNDER THE GUIDANCE OF

DR. AMANDEEP VERMA

SUBMITTED TO THE

Information Technology Department

University institute of Engineering and Technology , Panjab University Chandigarh

On behalf of IPD Summer Training, June 2020

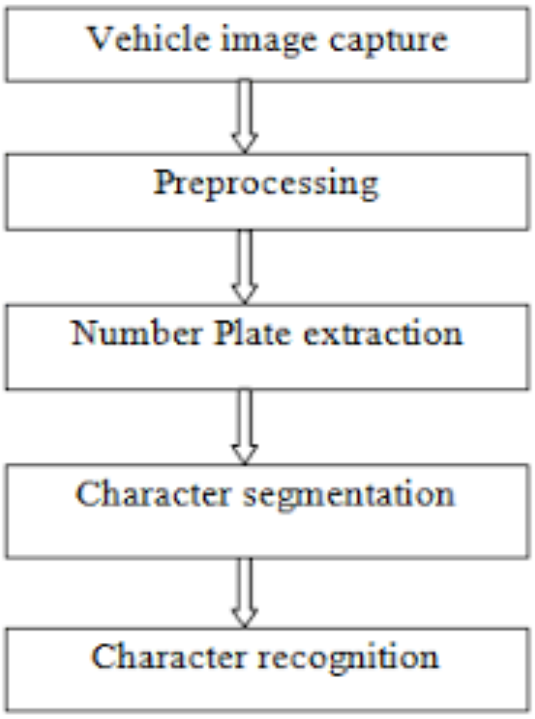
TABLE OF CONTENTS

Introduction::

- The purpose of this project is to read the vehicle number from photo of vehicle without human intervention. This project is very useful in many applications. This technique is called Automatic number plate recognition.(ANPR)

Uses::

- ANPR is used by police forces around the world for law enforcement purposes,including to check if a vehicle is registered or licensed.
- It is also used for electronic toll collection.
- One of the main applications of ANPR is parking automation and parking security ticketless parking fee management, parking access automation, vehicle location guidance, car theft prevention, "lost ticket" fraud, fraud by changing tickets,simplified, partially or fully automated payment process, amongs many others.
  - This technique is used for law enforcement in many countries like Australia, Belgium, Germany, England etc



CODE OF AUTOMATIC NUMBER PLATE DETECTION

***** There are some changes in python code file because code written below shows output side by side but this is not possible in case of python file. So to show image we use cv2.imshow function in that file.*****

---------- Start of code ----------

Import Libraries:

- First of all to make a project on python we need to import packages . so we import some of packages below.
- Numpy package is a python library used for working with arrays.
- Opencv(cv2) package is main package which we used in this project. This is image processing library.
- Imutils is a package used for modification of images . In this we use this package for change size of image.
- Pytesseract is a python library used for extracting text from image. This is an optical character recognition(OCR) tool for python.
- In matplotlib library we use a package name pyplot. This library is used for plotting the images. % matplotlib inline is used for plot the image at same place.It will not open image in another window.

In [1]:
```
import numpy as np
import cv2
import imutils
import pytesseract
import matplotlib.pyplot as plt
%matplotlib inline
pytesseract.pytesseract.tesseract_cmd = r"C:\Program Files\Tesseract-OCR\tesse
ract.exe"
```

imread function is used to read the image. It is defined in opencv library.

.

In [2]:
```
# Read the image file
image = cv2.imread('Car7.jpg')
```

Original Image

In this we define a function plot_image. This function will show the image with title without showing coordinates axis.

In [3]:
```
def plot_image(img1, title1=""):     # define a function to plot image
    fig = plt.figure(figsize=[15,15])
    ax1 = fig.add_subplot(121)
    ax1.imshow(img1, cmap="gray")
    ax1.set(xticks=[], yticks=[], title=title1)
```

Resize function is used to change the size of image .This is defined in imutils library.

In [4]:
```
image = imutils.resize(image, width=500)
```

In [5]:
```
# Display the original image
plot_image(image,"Orignal Image")
```

Original Image

Grayscale and blur image

cvtColor function is used to convert the color of image. In this we convert original image to grayscale image. This function is also defined in opencv library.

In [6]:
```
# RGB to Gray scale conversion
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
plot_image(gray,"Grayscale Conversion")
```

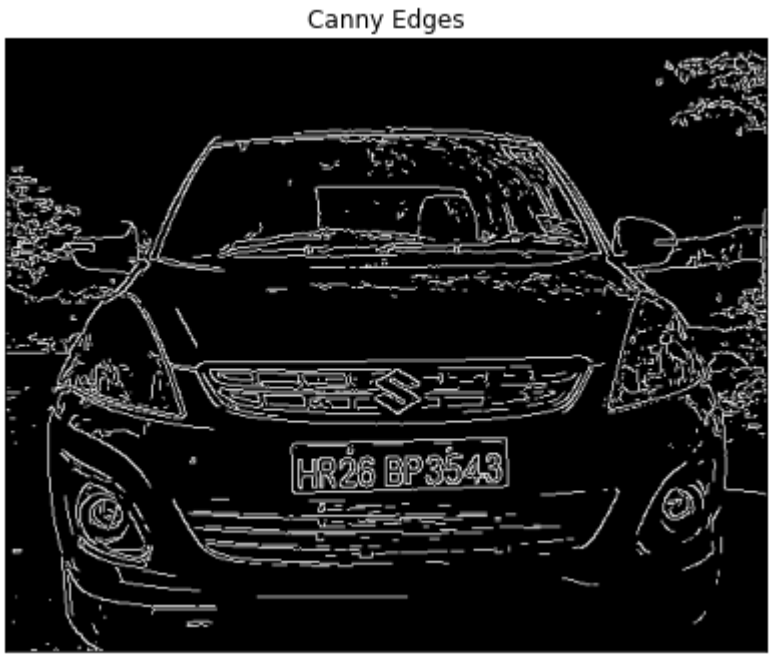
- bilateralFilter method is used to blur the image. This remove the noise from image/
- A bilateral filter is used for smoothening images and reducing noise, while **preserving edges**
- bilateralFilter is highly effective in noise removal while keeping edges sharp. But the operation is slower compared to other filters

In [7]:
```
# Noise removal with iterative bilateral filter(removes noise while preserving edges)
blur = cv2.bilateralFilter(gray, 11, 17, 17)
plot_image(blur,"Blur Image")
```

Canny Edge Detection

OpenCV puts all the above in single function, **cv.canny()** . This method is used for detection of edges of car. After appling this function image will be seen like a drawing on black paper with white color.

In [8]:
```
# Find Edges of the grayscale image
edged = cv2.Canny(gray, 170, 200)
plot_image(edged,"Canny Edges")
```



Canny Edges

- Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition.

- In OpenCV, finding contours is like finding white object from black background. So remember, object to be found should be white and background should be black.

Find Contours

- There are three arguments in **cv2.findContours()** function, first one is source image, second is contour retrieval mode, third is contour approximation method. And it outputs the image, contours and hierarchy. contours is a Python list of all the contours in the image. Each individual contour is a Numpy array of (x,y) coordinates of boundary points of the object.

- 1st argument is source image.

- 2nd argument is passed in this is **Contour Retrieval Mode**. We usually passed **cv.RETR_LIST** or **cv.RETR_TREE**

- Normally we use the **cv.findContours()** function to detect objects in an image, right ? Sometimes objects are in different locations. But in some cases, some shapes are inside other shapes. Just like nested figures. In this case, we call outer one as **parent** and inner one as **child**. This way, contours in an image has some relationship to each other. And we can specify how one contour is connected to each other, like, is it child of some other contour, or is it a parent etc. Representation of this relationship is called the **Hierarchy**.

- Third argument is Contour Approximation method.There are two methods in this one is CHAIN_APPROX_SIMPLE and another is CHAIN_APPROX_NONE

- If you pass cv2.CHAIN_APPROX_NONE, all the boundary points are stored. But actually do we need all the points? For eg, you found the contour of a straight line. Do you need all the points on the line to represent that line? No, we need just two end points of that line. This is what cv2.CHAIN_APPROX_SIMPLE does. It removes all redundant points and compresses the contour, thereby saving memory.

In [9]:
```
# Find contours based on Edges
cnts, new = cv2.findContours(edged.copy(), cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
```

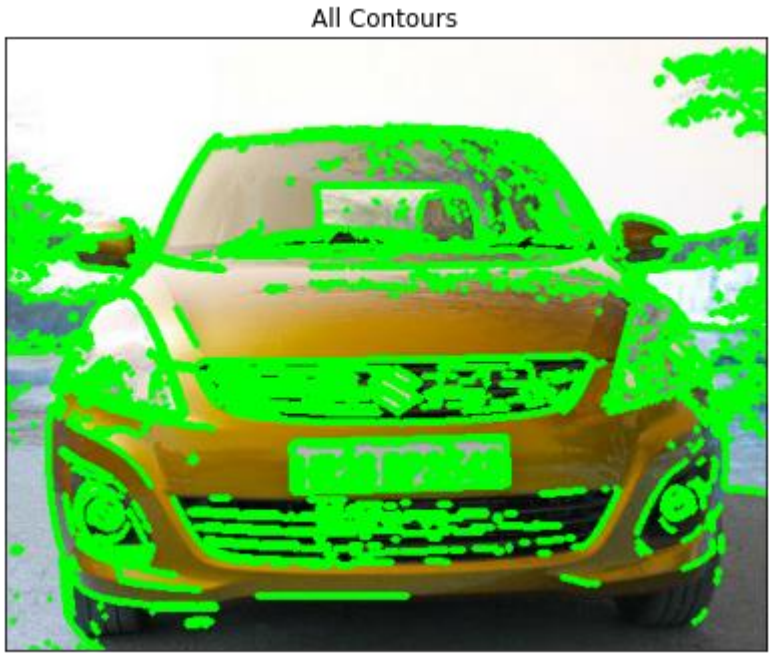
In [10]:
```
print(len(cnts))
```

```
981
```

Draw Contours

- To draw the contours, `cv2.drawContours` function is used. It can also be used to draw any shape provided you have its boundary points. Its first argument is source image, second argument is the contours which should be passed as a Python list, third argument is index of contours (useful when drawing individual contour. To draw all contours, pass -1) and remaining arguments are color, thickness etc.
- 1st argument is the source image . Notice that in this we create a copy of image because this function will change the image content so to preserve the original image we make a copy of this.
- 2nd argument is the list of contours.
- 3rd argument is value that contour we have to draw. If we pass -1 in this then this will draw all contours
- 4th argument is color in rgb format.
- 5th argument is the thickness of drawn contours.

In [11]:
```
# Create copy of original image to draw all contours
img1 = image.copy()
cv2.drawContours(img1, cnts, -1, (0,255,0), 3)
plot_image(img1,"All Contours")
```



Sort Contours

- Now we are sorting the contours in reverse order areawise .
- Cv2.contourArea method is used to find area of contours.
- We took first 30 countous out of 981 countous according to their area.
- This will decrease the time of ouput.

In [12]:
```
#sort contours based on their area keeping minimum required area as '30' (anyt
hing smaller than this will not be considered)
cnts=sorted(cnts, key = cv2.contourArea, reverse = True)[:30]
NumberPlateCnt = None #we currently have no Number plate contour
```

In [13]:
```
print(len(cnts))
```

```
30
```

Now we are showing first 30 contours . This will reduce the time of output .

In [14]:
```
# Top 30 Contours
img2 = image.copy()
cv2.drawContours(img2, cnts, -1, (0,255,0), 3)
plot_image(img2,"Top 30 Contours")
```

Finding number plate

- This is the main part of our project. Main logic is implemented in this loop.We loop through all the contours and find perimeter of all contours using arcLength() method.
- approxPolyDP method is used to detect the shape according to our requirements.
- In this number plate is rectangular in shape so we need rectangular coordinates.
- This will tell the coordinates of shape and we know rectangle has 4 sides so if length of this approx variable must be 4.
- When length is 4 and to make a rectangle we use boundingRect method
- It is a straight rectangle, it doesn't consider the rotation of the object. So area of the bounding rectangle won't be minimum. It is found by the function **cv.boundingRect()**.

In [15]:
```
# loop over our contours to find the best possible approximate contour of numb
er plate
count = 0
for c in cnts:
        peri = cv2.arcLength(c, True)
        approx = cv2.approxPolyDP(c, 0.02 * peri, True)
        if len(approx) == 4:
            NumberPlateCnt = approx #This is our approx Number Plate Contour

            # Crop those contours and store it in Cropped Images folder
            x,y,w,h=cv2.boundingRect(c) #This will find out co-ord for plate
            new_img = gray[y:y + h, x:x + w] #Create new image
            cv2.imwrite('contour image.jpg', new_img) #Store new image
            break
```

Cv2.imwrite function is used to create a new image

In [16]:
```
# Drawing the selected contour on the original image
print(NumberPlateCnt)
cv2.drawContours(image, [NumberPlateCnt], -1, (0,255,0), 3)
plot_image(image,"Final Image With Number Plate Detected")
```

```
[[[328 262]]

 [[190 265]]

 [[188 297]]

 [[326 294]]]
```

Final Image With Number Plate Detected

In [17]:
```
crop_img=cv2.imread('contour image.jpg')
plot_image(crop_img,"Cropped Image")
```

Cropped Image

- Pytesseract library is used to convert image text to string . There is a function name image_to_string defined in this library. This will convert our image text to string.
- And then using print method we print this on screen
- Python-tesseract is an optical character recognition (OCR) tool for python. That is, it will recognize and “read” the text embedded in images.

Character segmentation:

Character segmentation is an operation that seeks to decompose an image of a sequence of characters into subimages of individual symbols. It is one of the decision processes in a system for optical **character** recognition (OCR).

Optical Character Recognition(OCR) is a process which allows us to convert text contained in images into editable documents. **OCR** can extract text from a scanned document or an image of a document; really, any image with text in it.

In [18]:
```
# Use tesseract to covert image into string
text = pytesseract.image_to_string('contour image.jpg', lang='eng')
print("Number is :", text)
```

```
Number is : HR26 BP3543.
```

---------- End of code ----------

Conclusion and Future Scope:

This was great experience to make this project. We learned a lot from this project . This project can be extended to further level. Logic applied in this project can be make accurate using machine learning.

As a future work the developed system would be concentrated upon increasing the accuracy of text localization and graphics removal in caption textimages. It can be evaluated using various other available image databases and using various other classifiers.

References:

- www.python.org
- www.w3schools.com
- www.stackoverflow.com